Windows* 98 Latency Characterization for WDM Kernel Drivers

Revision 1.01
30 September 1998
Intel Architecture Labs
Intel Corporation

Erik Cota-Robles

1. Summary

This paper presents an initial latency characterization of WDM kernel driver latency on Windows* 98. We arrange the various kernel and user mode services made available by Windows into an operating system (OS) scheduling hierarchy and use this hierarchy to define a comprehensive set of metrics for quantifying the real-time performance of Windows. We collected latency data for each of the key levels in this OS scheduling hierarchy under a variety of application loads on a Pentium® II processor system configured to conform to the minimum system requirements of the *PC 99 System Design Guide* [PC99]. We identify two Windows 98 components (one optional, one easily disabled) and one subclass of applications that present challenges to the deployment of low latency drivers such as soft audio and softmodem and present data collected with and without these components and applications. We use this data to analyze the real-time performance of Windows 98 under load and project the quality of service that a softmodem should be able to realize under Windows 98.

OS latency distributions are highly non-symmetric, having a long tail and a single mode, with the result that the mean and standard deviation are not useful measures of the distribution. The preferred statistics for characterizing such skewed distributions are the median and the extremes. In order to give a comparative measure of the "thickness" of the tail, Table 1 presents the observed median as well as the hourly, daily and weekly expected maximum latencies for a PC 99 minimum System running both Business and Consumer Applications. Daily and weekly values reflect consumer and business usage patterns described in section 4. Large thread latencies for Business Applications come principally from Workstation CAD applications, while for Consumer Applications they come principally from multimedia (i.e., 3D Games and Real Audio).

	PC 99 Min. System Running Business Apps (5 day week, 7 hour day)				**PC 99 Min. System Running Consumer Apps (7 day week, 5 hour day)**			
OS Service	**Median**	**Expected Max / Hr.**	**Expected Max / Day**	**Expected Max / Wk.**	**Median**	**Expected Max / Hr.**	**Expected Max / Day**	**Expected Max / Wk.**
H/W Interrupt to S/W ISR	0.1	1	3.7	5.7	0.1	8	9.7	12.2
H/W Interrupt to DPC	0.1	1.4	4.2	6.3	0.1	8.6	11.8	14.3
H/W Interrupt to kernel RT thread (**High** Priority)	0.1	22.7	61.8	80	0.1	34.4	69.4	88
H/W Interrupt to kernel RT thread (**Def**. Priority)	0.2	22.9	68.3	81.9	0.2	35.1	74.7	89.9

Table 1: Observed Latency in Milliseconds to Windows 98 Kernel Services on a PC 99 System (Optional Virus Scanner not Installed, Sound Scheme Set to "None", VxD-only 3D Games Excluded)

2. Introduction

This paper is primarily intended for developers of Win32* Driver Model (WDM) kernel drivers and OEMs who are working to enable cost reduction via migrating processing to the host processor. This paper:

- Defines a comprehensive set of real-time performance metrics for Windows 98 and Windows NT*.
- Provides initial data on Windows 98 latencies on a PC 99 minimum systems.
  (300 MHz Pentium II processor with 512KB L2 cache, 32 MB system RAM, AGP graphics, PCI 2.2, no ISA devices).
- Documents latency measurement procedures.

Our goal in publishing this paper is to further joint work with Microsoft and industry leaders to identify OS-related and external factors that have impact on OS scheduling services.

2.1 Organization

This paper is organized as follows. The remainder of section 2 defines the OS scheduling hierarchy and presents the objectives and goals of the Intel Architecture Labs' (IAL) Scalable Platforms Initiative in relation to soft migration. Section 3 defines a set of latency metrics and briefly describes the measurement tools. Section 4 briefly describes the system hardware and software configuration and then presents a detailed description of the stress application loads as well as the usage models which guided the collection of latency data for each of the load types. Section 5 presents the results of our measurements. In section 6 we present our analysis and in section 7 we conclude.

2.2 OS Scheduling Hierarchy

We abstract the services provided by the Win32 Driver Model (WDM) into an *OS scheduling hierarchy* as shown below. Each level of the OS scheduling hierarchy is fully preemptible by the level(s) above it.

- Interrupt Service Routines (ISRs)
  Fully preemptible, execute at IRQLs from `DIRQL` through `HighLevel`
- WDM Deferred Procedure Calls (DPCs)
  Non preemptible, FIFO/LIFO queue, 3 priorities (`HighImportance`, `MediumImportance`, `LowImportance`)
- Kernel Mode Real-Time Priority Threads
  Fully preemptible, not timesliced, execute at Win32 priorities 16 through 31
- Kernel Mode Normal Priority Threads
  Fully preemptible, timesliced, execute at Win32 priorities 1 through 15

The ISR and Thread levels are fully preemptible internally, so any thread or ISR can be preempted by a higher priority one. The DPC level is non-preemptible internally, so a higher "Importance" DPC cannot preempt an already executing lower "Importance" DPC. However, `HighImportance` DPCs are enqueued at the front of the queue (i.e., LIFO), ensuring that they will be dequeued for execution before lower (or same) priority DPCs which have not yet been dequeued for execution.

Since Windows 98 retains support for Windows 95 VxD drivers they are, of course, still available alongside the WDM ones in Windows 98. We will not be concerned with these legacy VxD OS scheduling services in this paper, but will confine ourselves to the top 3 levels of the WDM OS scheduling hierarchy, ignoring the normal priority timesliced kernel mode threads because they do not support real-time computation on the host processor. For a full account of the Win32 Driver Model the reader is referred to [Baker1997].

2.3 Objectives and Goals

Among the objectives of the Intel Architecture Lab (IAL) Scalable Platforms Initiative is to support the successful migration of hardware functionality into host-based software by promoting the industry-wide diffusion of host-based signal processing technologies in order to enable cost-reduced platforms. IAL engineers are developing general real-time scheduling recommendations that can be applied to specific next-generation WDM kernel drivers, such as modem and audio, to ensure maximum performance and compatibility by design. The initial set of recommendations have been included in the Intel-Microsoft PC 99 Design Guide chapter on modems [PC99ModemChapter], which contains guidelines for the design of robust WDM driver-based software modems (i.e., modems with a data pump that executes on the host processor).

A key milestone on the road to achieving this objective is to document the timing behavior of key portions of the Windows OS scheduling hierarchy on PC 99 minimum systems. With the new Win32 Driver Model (WDM) Windows 98 shares a common functional architecture for kernel-mode device drivers with Windows NT. Although the Windows 98 and Windows NT driver architectures are functionally compatible, their timing behavior is quite different. In a future report we plan to characterize the differences in timing behavior between Windows 98 and Windows NT using the latency metrics which we develop and present in this report. Here we present the timing behavior of Windows 98. It is known that Windows 95 has uniformly longer latencies than Windows NT 3.51 and 4.0 across all levels of the OS scheduling hierarchy [Held1996][Cota-Robles1997] and preliminary data indicates that this continues to be the case with Windows 98. Thus the data on Windows 98 presented here should be useful to all developers of WDM software drivers.

By collecting and publishing detailed information on the timing behavior of Windows 98 and Windows NT under a variety of well-defined, repeatable loads, the Intel Architecture Lab (IAL) will help enable Independent Hardware Vendors (IHVs) and Independent Software Vendors (ISVs) to design more robust WDM kernel drivers by:

- Identifying fundamental latency characteristics of PC 99 minimum systems running Windows 98 and Windows NT while executing business productivity, consumer multimedia, and high-end CAD applications.
- Enabling audio, modem and other IHVs/ISVs to make more well informed feature vs. cost decisions based on well-characterized and reproducible timing behavior for legacy-free PC 99 systems.
- Establishing an industry-wide dialog regarding the real-time performance of Windows.

3. Latency

3.1 Motivation

Host-based signal processing drivers often perform significant amounts of hard real-time processing (e.g., the datapump for a software modem) in a manner that takes too long for an ISR or requires facilities such as the floating point coprocessor that may not be readily available in interrupt context. As a result such processing is typically implemented in kernel mode threads and/or WDM DPCs. A determining factor in whether such a driver will meet its deadline is the delay, or *latency*, between the time at which a hardware interrupt occurs and the time at which it is effectively serviced by DPC or thread-based processing. Since host-based signal processing data processing commonly involves handing off the data from task to task for processing, thread context switch time can also be a concern for drivers which use thread-based processing. In this investigation we focus on the interrupt and task latency metrics, which were first defined in [Cota-Robles1997] and [Held1996]. Together with thread context switch time, interrupt and task latency incorporate the essential overhead imposed by the operating system on real-time tasks.

Since a host-based signal processing driver task can be preempted and held off from completing a computation after it has begun executing, low latency is a necessary but *not* sufficient condition that a task will complete its computation by the applicable deadline. Windows software executes in an open environment with an unknown mix of other software executing concurrently. WDM DPCs and kernel mode threads in Windows 98 and Windows NT are both preemptible by software that executes at higher levels in the OS scheduling hierarchy. It is thus impossible to absolutely guarantee that a host-based signal processing driver task will not be preempted after execution has begun but before it completes its computation. An approximate analytical approach that yields reasonable results in practice is presented in [Cota-Robles1997]. The Intel Architecture Lab (IAL) is also developing a tool which models computation at various points in the OS scheduling hierarchy.

3.2 Interrupt Latency

Interrupt latency is defined to be the delay from the assertion of the hardware interrupt, *as seen by the processor*, until the first instruction of the software interrupt service routine (ISR) is executed. It thus measures the time to initial servicing of an interrupt, encompassing the maximum time during which interrupts are disabled as well as bus latency necessary to resolve the interrupt but does not include bus latency prior to the assertion of the interrupt at the processor.

DPC latency is defined to be the delay from the time at which the software ISR enqueues a DPC until the first instruction of the DPC is executed. Because ordinary DPCs queue in FIFO order, DPC latency encompasses the time required to enqueue and dequeue a DPC as well as the aggregate time to execute all DPCs in the DPC queue.

3.3 Thread Latency

Thread latency is defined to be the time from the setting of a WDM event, semaphore, etc. in a DPC or ISR to the first instruction after the Wait is satisfied in a thread that is waiting on the respective event, semaphore, etc. It thus measures the

worst case thread dispatch latency for a thread waiting on an interrupt, measured from the DPC or ISR itself to the first instruction executed by the thread after the wait. Thread latency encompasses a variety of thread types and priorities (e.g., kernel mode high real-time priority) and includes the time required to save and restore thread context, performance of semaphores and the maximum time during which the operating system disables thread scheduling.

3.4 Latencies Measured for This Paper

For this paper we present data for the following latencies:

- ISR latency for the Programmable Interval Timer (PIT) ISR.
- DPC latency for a "Medium Importance" WDM DPC enqueued by the PIT ISR. We term this DPC the PIT DPC.
- High priority kernel mode thread latency for a Win32 priority 28 kernel mode thread signaled by a WDM `SynchronizationEvent` set in the PIT DPC.
- Medium priority kernel mode thread latency for a Win32 priority 24 kernel mode thread signaled by a WDM `SynchronizationEvent` set in the PIT DPC.

3.5 Latency Measurement Tools

The latency measurement tools measure the delay from occurrence of a hardware or software event until the execution of the first instruction of a software entity which executes at one of the levels in the OS scheduling hierarchy (e.g., an interrupt service routine or ISR). The tools are implemented as WDM device drivers with command line interfaces and measure each of the latencies listed in the previous section. When the originating software entity executes, the time stamp register is read and the value saved in page-locked memory. When the destination software entity executes, the time stamp register is read again and the values differenced to yield the latency. The latencies are returned to the application via an IRP generated by a call to ReadFileEx.

In the case of the Interrupt Latency Driver the driver sets a timer to expire in a given number of milliseconds and reads the time stamp register. The driver then uses the time stamp register to calculate when in clocks the timer should expire and records the time stamp when the software ISR actually executes. The target software entity (in this case, the software ISR) then proceeds as above. This approach suffers from limited resolution (basically +/- the Programmable Interval Timer (PIT) frequency), which is adequate to resolve long interrupt latencies with reasonable accuracy.

4. Latency Measurement Procedure

Previous studies of the real-time performance of Windows focused on the use of Windows in dedicated environments where only a single application is running [Heubaum1995][Microsoft1995]. This is necessary but not sufficient information to evaluate the potential of Windows for host-based signal processing drivers and applications. Host-based signal processing drivers and applications are not standalone, dedicated applications. From the standpoint of the "lowest level" real-time software (e.g., kernel modem or low latency audio) the rest of the application (e.g., the user mode video codecs and GUI display of RealAudio Player) is for all practical purposes an external application load. Since user mode applications can be a noticeable impediment to timely response by the operating system, we measured latency in the presence of the stress from unrelated applications.

For each application category we estimated how many hours per week constitutes heavy use for applications belonging to that category. We used these estimates to calculate how many of hours of data we needed to collect using the various stress applications. Some of the stress applications were driven by Microsoft Test (MS-Test) or other factors at speeds much faster than those at which a typical heavy user would use them, enabling us to collect data over a shorter period of time than would otherwise have been the case. We collected data for periods of hours, as described below, and were thus able to capture events that occur at frequencies as low as 1 in 100,000 in statistically significant numbers.

4.1 Test System Configuration

Our goal was to characterize the OS response time which a kernel mode driver would encounter on a PC 99 compliant system. Thus, we configured our system exclusively with PCI and USB devices, as follows:

- Pentium II processor, 300 MHz w/ 512k cache
- Intel 440LX chipset

- 32 MB system RAM
- ATI* Xpert@Work* AGP graphics (1024x768 with 32-bit color; except for 3D games, 800x600 with 32-bit color)
- Maxtor* DiamondMax* 6.4 GB UDMA Hard Drive
- Philips* DSS 350 USB speakers
- Sony* CDU 711E 32x CD-ROM drive
- Intel EtherExpress Pro/100 PCI NIC (Web browsing only)

We installed Windows 98 base OS together with the Plus! 98 Pack with the exception of the optional virus scan which was not installed[1]. DMA was enabled on the hard drive and the CD-ROM drive. The ISA Plug and Play Enumerator and motherboard (ISA) audio devices were disabled in the Control Panel System Properties menu.

4.2 Office Applications

To represent this class of applications the Winstone* 97 Business automated benchmark was selected. The Winstone 97 benchmark is driven by MS-Test scripts and runs a number of business productivity applications spanning three categories of business computing:

- Database: Access* 7.0, Paradox* 7.0
- Publishing: CorelDRAW* 6.0, PageMaker* 6.0, PowerPoint* 7.0
- Word Processing and Spreadsheet: Excel* 7.0, Word* 7.0, WordPro* 96

Each application is installed via an Install Shield script, run at full speed through a series of typical user actions and then uninstalled.

MS-Test drives the applications at speeds in excess of human abilities to type and click a mouse. As a result the system is stressed more than would actually occur during normal usage and it is not necessary to collect data for as long a period of time as would otherwise be the case. Assuming approximately a 10 to 1 ratio of MS-Test input speed to human input speed, we conclude that Winstone 97 Business running continuously will produce as much system stress in 4 hours as a heavy user will produce in 1 work week (i.e., 40 hours). We collected data with several “sound schemes” because we observed that Windows 98 sound schemes, which associate sounds with user-visible “events” (e.g., closing a program), significantly impact observed thread latencies.

The test procedure was to launch Winstone 97, select the Business benchmark and then minimize the Winstone 97 application. The latency measurement tools were then launched, the Winstone 97 application maximized and the Business benchmark launched.

4.3 Multimedia Applications

We divided this class of applications into two subcategories: 3D games and Web browsing with enhanced audio/video. Specific applications were chosen, so unlike the Winstone benchmarks these applications must be driven manually or by custom MS-Test scripts. As a result we discuss each subcategory separately.

4.3.1 3D Games

Three 3D games were selected: Forsaken*, Freespace* Descent* and Unreal*. The latter two are very recently released games which run on both Windows 95 and Windows NT (indeed, Unreal advertises to run on Windows 98) and can thus be taken to be indicative of the performance to be expected in the presence of new games in the PC 99 time frame. Forsaken, on the other hand, is a Windows 95 only game and can be taken to be indicative of the performance to be expected in the presence of VxD games.

The games come with brief product demos which are designed to highlight graphic special effects and other multimedia features of the game and approximate the system load which a skilled game player can achieve. Thus, an hour running demos produces essentially the same amount of system load as an hour of playing the game, exclusive, of course, of any additional load produced by using a modem to play a multiplayer game. We estimate that game enthusiasts play on the order of 2 to 3 hours per day, 4 to 6 days per week, leading us to conclude that 12.5 hours of data will capture about 1 week of play, on average, by a game enthusiast.

[1] Initial characterization of latencies with virus scan installed indicate an increase of 2 to 3 orders of magnitude in the frequency of long thread latencies for high priority kernel mode threads. Intel is working with Microsoft to resolve this.

The test procedure was to launch the latency measurement tools and then launch each game and its demo in turn.

4.3.2 Web Browsing and Playing Audio and Video Clips

Web browsing is dominated by download times. With a modem on a POTS line a heavy user is bandwidth limited. By using a network connection, downloading occurs at speeds far in excess of those achievable on a POTS line. As a result the system is stressed more than would actually occur during normal usage and it is not necessary to collect data for as long a period of time as would otherwise be the case. Assuming at least a 10 to 1 ratio of 10 MBit Ethernet download speed to POTS download speed, we estimate an overall 4 to 1 ratio given that the user also spends time reading Web pages, listening to audio and video clips, etc. We estimate that a heavy user browses the Web about 3 to 4 hours per day. We conclude that 8 hours of data while browsing with an Ethernet network connection would capture about 1 week of web browsing over POTS by a heavy user.

The test procedure was to launch the latency measurement tools and then to browse with both Netscape* Communicator* and Internet Explorer* 4.0. A variety of file types are downloaded, viewed or otherwise touched and numerous RealAudio* and Shockwave* audio clips were played.

4.4 Workstation applications

To represent this class of applications the Winstone 97 Highend automated benchmark was selected. The Winstone 97 benchmark is driven by MS-Test scripts and runs a number of highend workstation applications spanning three categories of workstation computing:

- Mechanical CAD: AVS* 3.0, Microstation* 95
- Photoediting: Photoshop* 3.0.5, Picture Publisher* 6.0, P-V Wave* 6.0
- Software Engineering: Visual* C++ 4.1 Compiler

Each application is installed via an Install Shield script, run at full speed through a series of typical user actions and then uninstalled.

MS-Test drives the applications at speeds in excess of human abilities to type and click a mouse. As a result the system is stressed more than would actually occur during normal usage and it is not necessary to collect data for as long a period of time as would otherwise be the case. On the other hand, workstation applications are inherently more stressful than business applications, being CPU or non-user I/O bound (as opposed to waiting on user input) a much larger fraction of the time than business applications. We thus more conservatively assume a 5 to 1 ratio running continuously will produce as much system stress in 6 hours as a heavy user will produce in 1 work week, assuming that an engineer spends about 32 hours at his/her workstation.

The test procedure was to launch Winstone 97, select the Highend benchmark and then minimize the Winstone 97 application. The latency measurement tools were then launched, the Winstone 97 application maximized and the Highend benchmark launched.

5. Win98 OS Latency Data

5.1 Observed Latency Data

Windows OS latency distributions are highly skewed, having a long tail, and correspond in general to a “Chi-Squared” or “Gamma” distribution. Thus the mean and standard deviation are not particularly useful measures of the distribution. The preferred statistic of central tendency for this type of non-symmetric distributions, the median is < 0.1 ms. for the interrupt and DPC levels of the OS scheduling hierarchy, but ranges from 0.1 to 0.7 ms. for kernel real-time threads, depending on the thread priority and the application workload category. For brevity and clarity we omit it from the tables below.

For the purposes of forecasting realizable application worst case behavior we are primarily concerned with the worst case latency and with comparative measures of the “thickness” of the tail of the latency distribution. We therefore characterize the distributions in terms of three expected worst case values: hourly, daily and weekly. The hourly value is for continuous usage, whereas the daily and weekly do *not* represent continuous use for 24 or 168 hours, respectively, but rather expected average daily and weekly use by a heavy user. The usage patterns are described in detail in section 4, but briefly, for the

Office and Workstation applications a “day” represents 8 hours and a week has 5 “days”, while for the 3D games and Web Browsing a “day” represents 3 to 4 hours and a week has 7 “days”.

	Observed Hourly, Daily and Weekly Worst Case Windows 98 Latencies (in ms.)											
	Office Apps			Workstation Apps			3D Games (non-VxD)			Web Browsing		
OS Service	Max Per Hour	Max Per Day	Max Per Week	Max Per Hour	Max Per Day	Max Per Week	Max Per Hour	Max Per Day	Max Per Week	Max Per Hour	Max Per Day	Max Per Week
H/W Interrupt to S/W ISR	<1.0	1.4	1.6	2.2	5.6	6.3	8.8	9.7	12.2	1.1	1.7	3.5
S/w ISR to DPC	+ 0.1	+ 0.1	+ 0.4	+ 0.5	+ 0.5	+ 0.6	+ 0.9	+ 2.1	+ 2.1	+ 0.2	+ 0.3	+ 0.3
H/W Interrupt to DPC	1.0	1.5	2.0	2.7	6.1	6.9	9.7	12	14	1.3	2.0	3.8
DPC to kernel RT thread (High Priority)	+ 1.6	+ 5.2	+ 31	+ 21	+ 24	+ 24	+ 35	+ 46	+ 70	+ 14	+ 68	+ 80
H/W Interrupt to kernel RT thread (High Priority)	2.6	6.7	33	24	30	31	45	58	84	15	70	84
DPC to kernel RT thread (Medium Priority)	+ 3.1	+ 6.7	+ 31	+ 21	+ 23	+ 24	+ 36	+ 47	+ 70	+ 51	+ 68	+ 80
H/W Interrupt to kernel RT thread (Med. Priority)	4.1	8.2	33	24	29	31	46	59	84	52	70	84

Table 2: Windows 98 Interrupt and Thread Latencies with no Sound Scheme and with only non-VxD 3D Games on a PC 99 Minimum System

	Observed Hourly, Daily and Weekly Worst Case Windows 98 Latencies (in ms.)											
	Office Apps			Workstation Apps			3D Games			Web Browsing		
OS Service	Max Per Hour	Max Per Day	Max Per Week	Max Per Hour	Max Per Day	Max Per Week	Max Per Hour	Max Per Day	Max Per Week	Max Per Hour	Max Per Day	Max Per Week
H/W Interrupt to S/W ISR	<1.0	1.6	1.8	2.5	4.5	5.6	19	22	29	1.1	1.7	3.5
S/W ISR to DPC	+ 0.2	+ 0.3	+ 0.4	+ 0.5	+ 0.5	+ 0.6	+ 0.9	+ 2.1	+ 2.1	+ 0.2	+ 0.3	+ 0.3
H/W Interrupt to DPC	1.1	1.9	2.2	3.0	5.0	6.2	20	24	31	1.3	2.0	3.8
DPC to kernel RT thread (High Priority)	+ 36	+ 67	+ 71	+ 21	+ 24	+ 24	+ 63	+ 73	+ 86	+ 14	+ 68	+ 80
H/W Interrupt to kernel RT thread (High Priority)	37	69	73	24	29	30	83	97	117	15	70	84
DPC to kernel RT thread (Medium Priority)	+ 41	+ 69	+ 71	+ 22	+ 25	+ 26	+ 63	+ 73	+ 86	+ 51	+ 68	+ 80
H/W Interrupt to kernel RT thread (Med. Priority)	42	71	73	25	30	32	83	97	117	52	70	84

Table 3: Windows 98 Interrupt and Thread Latencies with the Default Sound Scheme and Including VxD 3D Games on a PC 99 Minimum System

Table 2 and Table 3 present the observed hourly, daily and weekly worst case interrupt, DPC and thread latencies in milliseconds for each of the application categories. DPC latencies are given twice: first as measured from the software ISR to the DPC (shaded), and immediately below from the H/W interrupt to the DPC (unshaded). The latter is inferred by adding the corresponding worst case times. Similarly for thread latencies, except that the shaded figures are measured from the DPC to the thread.

The system is a PC 99 minimum system running Windows 98 with all components of the Plus! 98 Pack installed *except* for the Plus! 98 virus scanner, an optional component which seriously impacts thread latency (see Figure 3, below). Table 2 presents data with a "no sound" sound scheme (i.e., sound "(None)" selected for all events) and with only non-VxD 3D games that run on both Windows 98 and Windows NT included. Table 3 presents data for the same system configured with the default sound scheme and with VxD 3D games that run only on Windows 98 included. We discuss these optional components as well as the 3D games in more detail in section 6, but note here that the impact is principally confined to the Office Apps and 3D Games categories.

6. Analysis

6.1 Anomalous Interrupt Latencies

Window 98 exhibited long interrupt latencies when running both the 3D Games and the Workstation Applications. These are principally caused by application VxDs and are all included in Table 3. Table 2 excludes the 3D game Forsaken which does not run on Windows NT in order to highlight the improvement in latency behavior that can be expected as increasing numbers of applications and drivers migrate to using WDM services. We explore this topic in more detail in section 6.1.2.

6.1.1 Workstation Interrupt Latencies

The Workstation CAD application "AVS 3.0" exhibits a single long worst case interrupt latency (3.0 ms. < x < 7.0 ms.). Because the duration of this latency is substantially less than the frequency at which our test polls to measure interrupt latency, occurrences of this latency are only observed sporadically, but it is very reproducible both as to magnitude and as to when during the Winstone 97 benchmark it occurs. While longer than desirable, this latency will not pose a significant barrier to current soft drivers such as software modems or low latency audio because of the relatively short duration.

6.1.2 3D Game Interrupt Latencies

When under load with all of the three 3D game applications Windows 98 exhibits very long interrupt latencies (up to 30 ms., but generally between 5 and 20 ms.). The oldest game, Forsaken, causes Windows 98 to exhibit these long latencies much more frequently than the newer games, Free Space and Unreal. With Forsaken an interrupt latency of at least 5 ms. occurs on average about once every 3 minutes and one of at least 10 ms. occurs about once every 6 minutes. In contrast, with Unreal (the newest game) we observed only a single interrupt latency over 5 ms. in over 5 hours of continuous play.

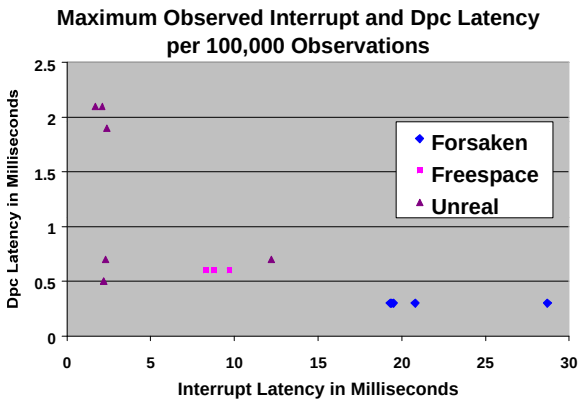

Figure 1: Scatter Plot Showing Inverse Correlation Between DPC Use and Worst Case Interrupt Latency

Clearly latencies of this magnitude will have a significant impact on soft modems, but they probably do not occur frequently enough to be of concern to soft audio. The exact cause of these latencies is under investigation, but they appear to be inversely correlated with the amount of DPC processing that the game does, as shown in Figure 1. It is worth noting in this context that unlike the other two games tested, Forsaken does not run on Windows NT and can thus be presumed to contain legacy VxD code which executes at interrupt context. It is our belief that as 3D game writers update their implementations to use WDM kernel mode services such as DPCs, interrupt latency will be reduced. A non-linear regression (exponential curve) on the data in Figure 1 indicates that about 75% of the variation can be explained by the hypothesis that use of DPCs is inversely correlated with

use of non-preemptible VxDs. This would tend to indicate that the frequency of observed long interrupt latencies, and to a lesser extent their magnitude, will come down relatively quickly as VxD based 3D games become less common.

6.2 Anomalous Thread Latencies

Two specific reproducible anomalies were noted which systematically increase the expected worst case thread latencies given in Table 1 and Table 2, principally for the non-multimedia workloads. Due to their nature they are likely to occur during normal use and are thus not properly classified as outliers. The virus scanner is an optional Windows component and it was therefore not installed on our test system. The sound schemes are integral to Windows and principally affect thread latencies; they have been excluded from Table 1 and Table 2, but are included in Table 3. We present log plots here showing the effect of including these features on the high priority real-time thread latency of Windows 98 when running the Winstone 97 Business benchmark suite ("Office Apps" in the tables).

6.2.1 Windows 98 Sound Schemes

Windows 98 enables the user to associate sounds with user-visible events and to group these associations into "sound schemes". We found that use of these sound schemes has a strong impact on thread latency regardless of priority level. For high priority real-time threads the frequency of observed long thread latency increases by an order of magnitude when the Windows default sound scheme is used as shown in Figure 2. The impact on the amount of buffering required to achieve a given mean time to failure is generally greater than an order of magnitude due to the slopes of the latency curves. For example, with no sound scheme a 5 ms. buffer will be adequate for a 1 in 100,000 failure rate, whereas with the default sound scheme about 65 ms. of buffering will be required for similar reliability. For example, for a softmodem datapump with a 16 ms. cycle time this translates into a requirement for five 16 millisecond buffers in order to guarantee an average of 20 minutes between buffer underruns when using the default sound scheme. In contrast, two 16 ms. buffers are more than sufficient to guarantee an average of 40 minutes between buffer underruns when not using a sound scheme.

6.2.2 Microsoft Plus! 98 Pack Virus protection

Microsoft Plus! 98 Pack includes a virus scanner that causes an increase of 2 to 3 orders of magnitude in the frequency of long thread latencies for high priority kernel threads as shown in Figure 3. These latencies are manifested during file copying and file I/O which involves the virus scanner. With typical settings for the Virus Scanner in place, downloading a file would likely cause a softmodem with a datapump implemented in kernel mode threads to retrain or possibly hang up. (This would *not*, however, apply to a modem with a datapump implemented as one or more WDM DPCs.) Similarly, opening a word document would cause breakup in a soft audio playback. If the virus scanner is disabled or not installed then a 5 ms. buffer will be adequate for a 1 in 100,000 failure rate, whereas with the default sound scheme about 65 ms. of buffering will be required for similar reliability. For example, given an audio stream using 10 ms. buffers (e.g., using Kmixer*) five 10 millisecond buffers will be required in order to guarantee an average of 17 minutes between buffer underruns at the renderer when the virus scanner is continuously active. In comparison, two 10 millisecond buffers are more than sufficient to guarantee an average of 35 minutes between buffer underruns at the renderer when the virus scanner is disabled.

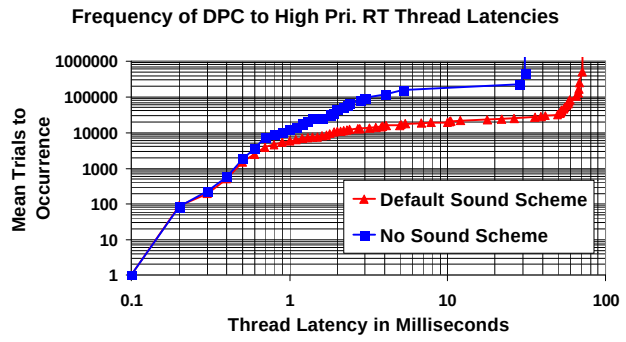

Figure 2: Effect of the Windows Default Sound Scheme on High Priority Real-Time Thread Latency

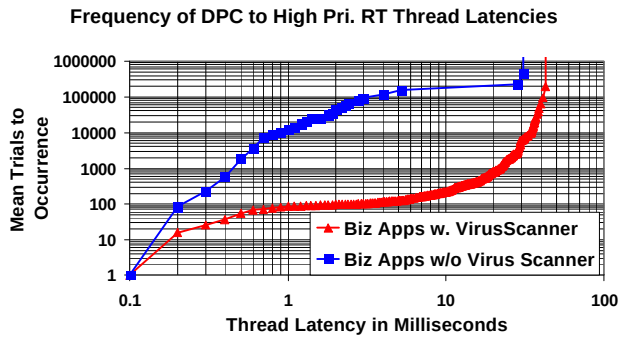

Figure 3: Effect of the Virus Scanner on High Priority Real-Time Thread Latency

6.2.3 Miscellaneous Outlier Latencies

Several reproducible pathological latencies were deemed to be outliers and excluded from consideration. The Forsaken demo mode exhibits pathologically long interrupt latencies (>100 ms.) upon autotermination of demo mode. These were discounted as not being reflective of what would occur during actual play.

In addition, the mechanical CAD application "Microstation" exhibits a pathologically long thread latency (50 to 70 ms.) during initial product registration. Because of the structure of the Winstone benchmark this occurs during every run. We discarded these latencies as being not reflective of the behavior that a user would encounter during normal use. However, it must be noted that users do, for example, download applications (e.g., Adobe Acrobat reader) and install them while running a modem, so a soft modem will not be able to completely ignore this usage scenario.

6.3 Softmodem Quality of Service

We conclude with a preliminary analysis of softmodem quality of service. Figure 4 shows the mean time to failure (i.e., buffer underrun) for the datapump of a softmodem as a function of the amount of buffering in the datapump. We have postulated that the datapump requires 25% of a PC 99 minimum system (300 MHz Pentium II processor) during data transmission mode, which is a conservative estimate. To interpret the diagram, calculate the total buffering in the datapump. For example, for a triple buffered implementation using 6 millisecond buffers, we can see that with 12 milliseconds of buffering the datapump will miss a buffer roughly once every 12 to 15 minutes while playing an "average" 3D game. With 10 millisecond buffers triply buffered, however, the datapump would average an hour between mises while playing an "average" 3D game. Note that missed buffers need not be catastrophic since design techniques exist whereby the datapump can arrange for hardware to automatically transmit a dummy buffer in the event that a buffer is missed. Such a dummy buffer can be made indistinguishable from line noise or other bit errors to the receiving modem and will result in either a retransmission request at the link layer or a dropped frame. In either case the overall impact on modem connection quality can be kept manageably small relative to the number of buffers which will be damaged due to line noise.

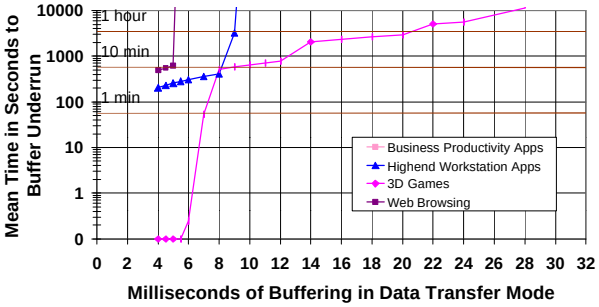

Figure 4: Mean Time to Failure for a DPC-based Datapump of a Softmodem on Windows 98 in Data Transfer Mode

7. Conclusions

We have characterized the latency behavior of the Windows 98 WDM OS scheduling hierarchy using the interrupt and task latency metrics first presented in [Held1996] and [Cota-Robles1997]. We have also developed a comprehensive usage model and used it to guide our selection of workload applications. Both the usage model and the latency metrics will be used in future reports on Windows NT 4.0 and, when it is released, 5.0.

Although the latency behavior of Windows 98 is significantly improved over that of Windows 95 there are still a number of challenges for developers of host-based drivers for audio, modem and other host-based real-time drivers. Chief among these are the 3D games, which impose very heavy graphics demands on the entire system, compromising the reliability with which the OS responds promptly to interrupts. Our analysis indicated that a requirement for VxD based 3D games to run concurrently with high reliability host-based real-time drivers such as softmodems could pose a significant challenge to developers. In contrast, newer 3D games which use the WDM DPC mechanism to accomplish their high priority processing exhibit a marked reduction in generated interrupt latencies and should be able to operate concurrently with a softmodem. Furthermore, based on our preliminary analysis we conclude that a soft datapump implemented as one or more DPCs should be feasible if the datapump has about 16 milliseconds of buffering. It is possible that the presence of VxD drivers in the Windows 98 environment will constrain the ability of implementers to deliver high reliability drivers which are more time critical than softmodems, whether those solutions are implemented as DPCs or as ISRs.

Thread-based drivers will also be impacted by Windows "sound schemes" and the virus scanner from the Microsoft Plus! Pack. Either of these can significantly constrain Windows' ability to promptly dispatch high priority real-time threads. Intel is working with Microsoft on these issues. Even with both of these workarounds in place, multimedia application loads (e.g., 3D games, Web browsing with streaming audio, etc.) will still pose significant challenges to host-based drivers for high reliability drivers such as soft modems. Audio drivers, particularly playback, will be less affected because of the greater latency tolerance of audio and the fact that sounds from the sound scheme interrupt the audio playback anyway.

8. References

[Baker1997] *The Windows NT Device Driver Book*. Baker, Art. Prentice Hall, Upper Saddle River, NJ. 1997.

[Cota-Robles1997] Schedulability Analysis for Desktop Multimedia Applications: Simple Ways to Handle General-Purpose Operating Systems and Open Environments. Cota-Robles, E., J. Held, T. J. Barnes. IEEE International Conference on Multimedia Computing and Systems, June 1997.

[Held1996] Real-Time Performance in Microsoft Windows Environments. Held, J., E. Cota-Robles, J. Jeyaseelan, R. Ramaswamy and S. Kambhatla. Intel Media & Interconnect Technology Lab Technical Report, February 1996.

[Heubaum1995] Windows 95 Interrupt Latency. Heubaum, Karl. Vireo White Paper, Vireo Software Inc., Acton, MA. September 1995.

[Microsoft1995] Real-Time Systems and Microsoft Windows NT. Microsoft Development Library White Paper, June 1995. Also available as http://www.microsoft.com/winntdev/realtime.htm , January 1996.

[PC99] PC 99 System Design Guide: A Technical Reference for Designing PCs and Peripherals for the Microsoft Windows Family of Operating Systems. Intel Corp. and Microsoft Corp., 1998. URLs: http://developer.intel.com/design/desguide/ and http://www.microsoft.com/hwdev/pc99.htm .

[PC99ModemChapter] PC 99 System Design Guide: A Technical Reference for Designing PCs and Peripherals for the Microsoft Windows Family of Operating Systems, Chapter 19. Intel Corp. and Microsoft Corp., 1998. URLs above.